1

TASK: UT20
CDRL: 04014
12 June 1992

UT20—PCTE Browser Tool User Manual Version 0.1

Informal Technical Data

STARS-TC-04014/003/00
12 June 1992

TASK: UT20
CDRL: 04014
12 June 1992

USER MANUAL
For The
SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

PCTE Browser Tool (PBT)
Version 0.1
SunOS Implementation

STARS-TC-04014/003/00
12 June 1992

Data Type: A005, Informal Technical Data

CONTRACT NO. F19628-88-D-0031
Delivery Order 0008

Prepared for:

Electronic Systems Division
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

Prepared by:

Paramax Systems Corporation
Tactical Systems
12010 Sunrise Valley Drive
Reston, VA 22091

TASK: UT20
CDRL: 04014
12 June 1992

USER MANUAL
PCTE Browser Tool (PBT)
Version 0.1
SunOS Implementation

Principal Author(s):

Michael J. Horton, Paramax, Valley Forge Labs *Date*

Approvals:

Thomas E. Shields 6/17/92

Task Manager *Dr. Thomas E. Shields* *Date*

(Signatures on File)

TASK: UT20
CDRL: 04014
12 June 1992

USER MANUAL
PCTE Browser Tool (PBT)
Version 0.1
SunOS Implementation

Change Record:

Data ID	*Description of Change*	*Date*	*Approval*
STARS-TC-04014/003/00	Original Issue: PCTE Browser Tool (PBT) user manual corresponding to PBT SunOS implementation, version 0.1 (RE: STARS-SC-03714A/001/00)	12 June 1992	*on file*

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
12 June 1992

3. REPORT TYPE AND DATES COVERED
Informal Technical Report

4. TITLE AND SUBTITLE
PCTE Browser Tool (PBT) User Manual
Version 0.1

5. FUNDING NUMBERS
F19628-88-D-0031

6. AUTHOR(S)
Paramax Systems Corporation

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Paramax Systems Corporation
12010 Sunrise Valley Drive
Reston, VA 22091

8. PERFORMING ORGANIZATION REPORT NUMBER
STARS-TC-04014/003/00

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Department of The Air Force
Headquarters Electronics Systems Division
Hanscom AFB, MA 01731-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
04014

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Distribution "A"

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The PCTE Browser Tool (PBT) is an instance of the Paramax STARS Reusable Graphical Browser (RGB), a generic graphical browser for the display of networks of nodes and arcs. In the case of the PBT, the nodes displayed by the RGB are PCTE *objects*, and the arcs are PCTE *links*. The PBT is an X Window System application, and as such makes use of the Paramax STARS implementation of Ada/Xt and Ada implementation of some MIT Athena and Hewlett Packard widgets.

14. SUBJECT TERMS
PCTE, Browser Tool, Ada Bindings

15. NUMBER OF PAGES
28

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
SAR

Contents

List of Figures

1 INTRODUCTION

1.1 SCOPE

The PCTE Browser Tool (PBT) is an instance of the Paramax STARS Reusable Graphical Browser (RGB), a generic graphical browser for the display of networks of nodes and arcs. In the case of the PBT, the nodes displayed by the RGB are PCTE *objects*, and the arcs are PCTE *links*. The PBT is an X Window System application, and as such makes use of the Paramax STARS implementation of Ada/Xt and Ada implementation of some MIT Athena and Hewlett Packard widgets.

This tool has been developed in Ada using the Sun Ada 1.0 compilation system from Sun Microsystems.

The PBT is ultimately intended for use in an ECMA PCTE environment, and has been implemented using the ECMA-162 Ada programming bindings to PCTE. However, in the absence of a conforming ECMA PCTE implementation, the PBT has been built on top of the Emeraude V12.2 PCTE 1.5 implementation, using a subset implementation of the ECMA Ada binding developed by Paramax STARS (release 0.1).

This document describes the general capabilities of the PBT and how to invoke them. For detailed instructions on how to install or rebuild the PBT, consult the associated Version Description Document (VDD).

1.2 PBT Overview

The PBT is designed to graphically display selected parts of a PCTE object base. Objects in the object base and the relationships amongst these objects are displayed at the PBT user's request. The PBT is intended to complement text-oriented commands such as `obj_list_links` and `obj_list_attr` that are included with the Emeraude V12.2 PCTE release—commands intended to be invoked from the text-oriented **esh** command shell.

The PBT allows the user to create and display any number of networks—*Views* in PBT terminology—of objects from a single PCTE object base. Figure 1 illustrates one such View.

In Figure 1, two basic kinds of objects are represented, using two different icons: PCTE *File* objects use a filled square as an icon, and ordinary PCTE objects without contents use a filled diamond inside an unfilled square as an icon. Recognizing that most objects in the PCTE object base are specializations of one of the basic PCTE object types, the PBT labels each object with the name of its object type as defined in one of the SDSes in the current PCTE *Working Schema*. It also labels each arc with the link name and key value associated with that PCTE link. The PBT distinguishes amongst the various kinds of links by decorating the arcs with different icons for different link kinds. In Figure 1, *Composition* links are labelled with *C*'s, *Reference* links with *R*'s, and *Implicit* links with *I*'s. When run

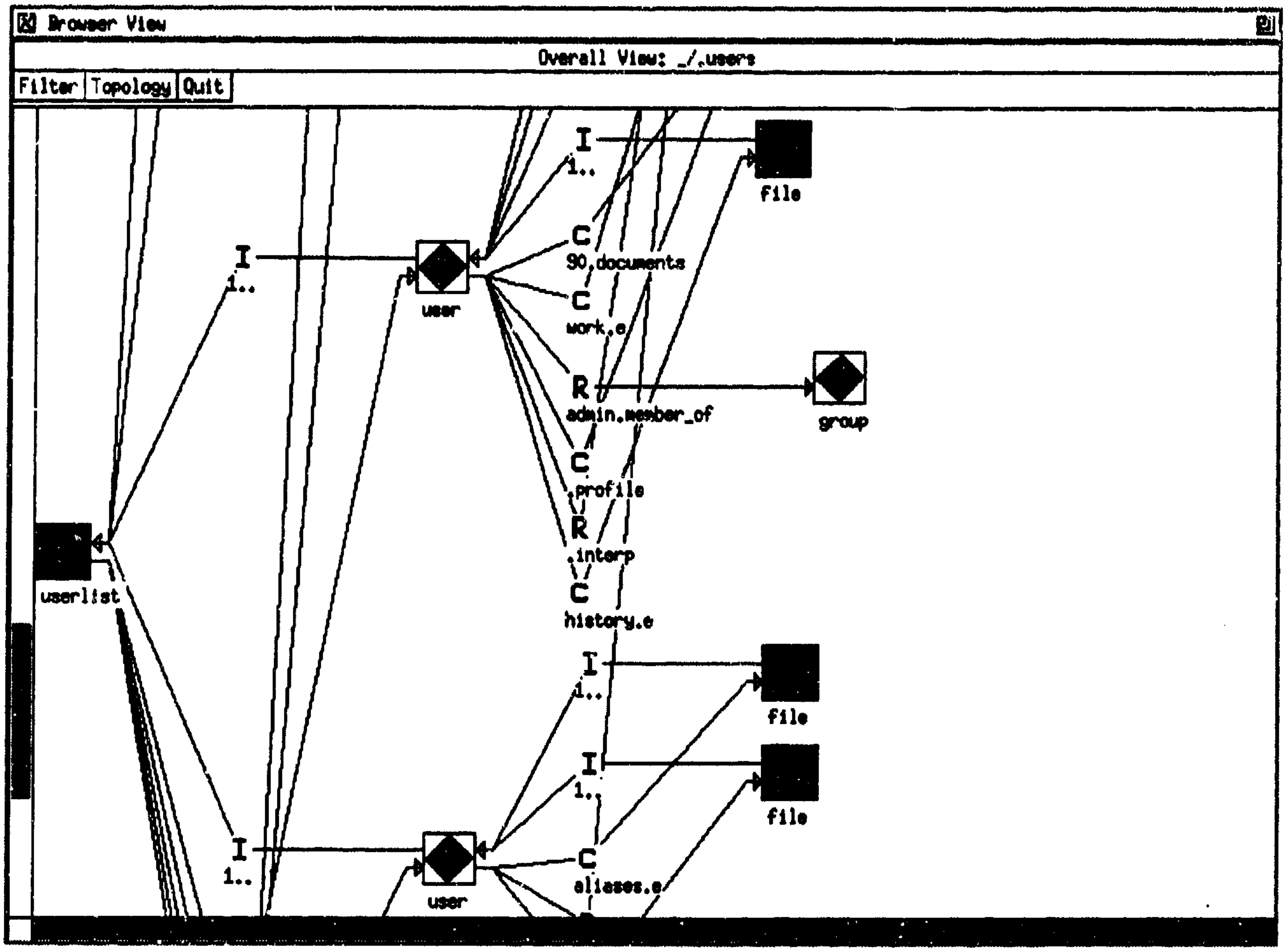


Figure 1: Graph of PCTE Objects and Links

on color monitors, the PBT may use different colors for the different node and arc icons to further distinguish amongst the different basic node and arc kinds.

The PBT supports two types of browser *Views*:

- Overall View
- Local View

An *Overall View* includes the set of PCTE objects making up the *Composition Tree* rooted at a specified PCTE object, i.e., which comprise the tree formed by traversing the PCTE *Composition* links emanating from the specified root object. At the user's discretion, an Overall View may also include other types of PCTE links (e.g., *Reference* links) emanating from the objects in the Composition Tree, together with the objects at which these links are targeted. In an attempt to limit the number of nodes in any Overall View to a "manageable" number, the PBT limits the tree depth to a user-controllable maximum tree limit.

A *Local View* consists of the set of PCTE objects which fall within a user-specified distance from a specified *start* object. The distance from the start object is defined as the minimum number of link traversals beginning at the start object to reach another object in the set. Local Views also include all of the links amongst the objects in the set.

The objects to be used as the roots of the composition trees for an Overall Views can be identified by the user via their PCTE pathnames. In addition, once Overall Views exist, root nodes of new Overall Views can be selected from amongst the already displayed nodes within these Views via point-and-click operations using the mouse. This allows the user to continue down the composition tree for branches within Overall Views that were cut off because of the maximum tree depth limitation controlling the Overall View construction.

The objects used as the starting point for new Local Views are always selected from amongst already displayed objects within existing Overall or Local Views via point-and-click operations.

The PBT supports a number of operations on nodes and links displayed in Views:

For every kind of PCTE object, it is possible to:

- display the object's pathname within that View.
- display its attributes.
- make it the PBT session's *Current Object.*
- create a new Overall View rooted at the node.
- create a new Local View starting from the node.

In addition, for a PCTE *File*, it is possible to:

- display its contents.

For every kind of PCTE link, it is possible to:

- display its full "linkname," defined as the pathname within that View of the link's target node.
- display the link attributes (if any).
- "go to" the source of the link by scrolling the View so that the source node is approximately centered within the View window.
- "go to" the target of the link by scrolling the View so that the target node is approximately centered within the View window.

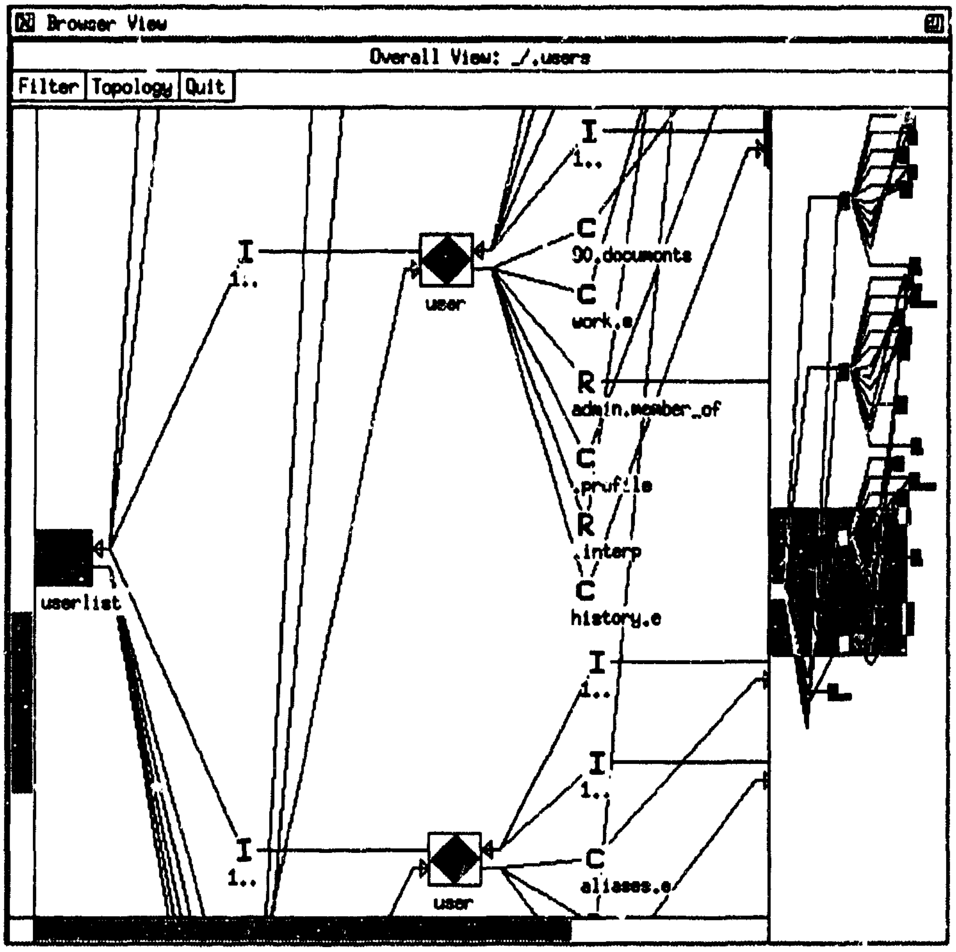


Figure 2: Topology Region Within View

Selected PBT operations require the user to identify PCTE objects via their pathnames. Such pathnames can start at either the root of the object base ("_") or the PBT user's home ("~"). Such pathnames can also be relative to the PBT session's *Current Object.* In addition, such pathnames can make use of user-defined object *aliases.* The user may define new aliases (which are local to a PBT session), or modify or delete existing ones. The PBT also supports the saving of such aliases in PCTE Files, and the loading of aliases from such Files.

Since the network of nodes in any particular View may be too large to fit within the window, the PBT supports mechanisms for scrolling around the View. The PBT also allows the user to attach a *Topology* region to any existing View—a region that contains a miniaturized depiction of the network, sometimes called the "view from one thousand feet." Figure 2 illustrates the same network displayed in Figure 1, with the Topology region now added to the right portion of the View.

The PBT supports a number of mechanisms by which one can filter out nodes and links from the networks displayed within Views.

Prior to the creation of Views, the user can control which links are to be included in such Views in two ways: via the PCTE *Working Schema* and via user-specified link name patterns.

The PBT allows the user to modify which SDSes are to be included in the PBT session's PCTE Working Schema. Links whose names are not defined in this Working Schema will not be displayed in Views. In addition, the PBT allows the user to specify one or more link names or link name patterns, and the PBT further limits the displayed links to be those matching such name patterns. Link name patterns can include the UNIX shell-like wildcards "*" and "?" (e.g., "`*list`"). The PBT allows the user to save these patterns in PCTE Files, and to load previously saved patterns from such Files.

For networks of nodes displayed in existing Views, the PBT supports the hiding and reexposing of nodes and/or links within these Views: PCTE objects of a specific kind (i.e., all PCTE Files or all non-file objects) can be hidden or exposed, as can links of a specific kind (e.g., all Composition, Reference or Implicit links). The PBT can also hide or expose objects based upon the names of their object types, and can hide or expose links based upon their link names.

The PBT allows the user to control some aspects of the PBT behavior, such as:

- Whether SDS names are to be used in Views within node and link labels.
- The maximum depth of Composition Trees associated with Overall Views.
- Whether to display PCTE exception information when PCTE exceptions are raised.

2 References

The following documents are applicable to the PCTE Browser Tool:

- PBT Version Description Document, Version 0.1, 12 June, 1992.
- Standard ECMA-149, Portable Common Tool Environment (PCTE) Abstract Specification, December, 1990.
- Emeraude V12, release 2, System Administration, July, 1991.
- Emeraude V12, release 2, Environment Guide, June, 1991.
- Emeraude V12, release 2, Tool Catalogue, June, 1991.
- Quercia, Valeria and O'Reilly, Tim, X Window System User's Guide for X11 R3 and R4, Volume 3, 3rd edition, O'Reilly & Associates.
- Fulton, Jim, X Window System, Version 11 Release 4 Release Notes, (supplied with MIT distribution)
- RGB User Manual, STARS-SC-03715/003/00, 5 September, 1991.

3 INSTRUCTIONS FOR USE

3.1 PREPARATIONS FOR USE

In order to use the PBT, the following preliminary steps must be taken:

1. The PBT executable must be installed in the environment so that it will be accessible by the user from within PCTE.
2. The set of files describing the X Window System resources and bitmaps to be used by the PBT must be installed in the UNIX file system, and the location of these files must be identified via an environment variable, using a full UNIX pathname.
3. The X Window System, X11R3 or X11R4, must be running on the user's terminal.
4. The server process for the PCTE object base must be running.
5. The user must log into PCTE.

3.1.1 Installing the PBT Executable

In the Emeraude implementation of PCTE, an executable such as PBT can either be installed as a "static context" within the PCTE object base itself, or can be moved into one of the UNIX directories within the UNIX path in affect during a PCTE session.

3.1.2 Installing the X Resource Files

A number of UNIX files associated with the PBT must be on-line at the time that the PBT is executed:

- A set of files describing the bitmaps to be used for the various node and link icons.
- The "X resource file" associated with the PBT, `PCTE-Browser`, describing such information as which bitmap to use for which type of object, what dimensions to use for the various widgets used by the PBT, etc.

These files are part of the PBT delivery. See the accompanying VDD for a description of how to install these files.

The PBT is typical of X-oriented tools in that the characteristics of the tool can be set or overridden by the user, rather than being hard-wired into the tool. These characteristics include such things as which bitmaps to use for specific types of node, or what physical display dimensions to use for the various widgets used by the browser. These characteristics ("resource values") are read by the tool each time it is started, and therefore, can be customized for each site or for individual users of the PBT. Appendix A contains the color monitor-oriented X resource file supplied with the PBT release—one which could be customized at any local installation of the PBT.

3.1.2.1 Identifying the X Resource Files

The PBT expects to read its X resource specification, **PCTE-Browser**, each time that the browser is invoked. The means by which X applications in general, and the browser in particular, expects to find such a file is via the **XAPPLRESDIR** environment variable. That is, the browser expects to find the newly installed **PCTE-Browser** file in the UNIX directory identified by the **XAPPLRESDIR** environment variable. The following command line ould be added to the **.login** or **.cshrc** file of every potential user of the browser, or else could be executed prior to invoking the PBT:

```
setenv XAPPLRESDIR directory-containing-PCTE-Browser
```

where *directory-containing-PCTE-Browser* is to be replaced by the site-specific location (i.e., UNIX pathname) of the UNIX directory containing the **PCTE-Browser** file—specifying the full UNIX pathname.

IMPORTANT NOTE: There are two versions of the PCTE X Resource File delivered with the PBT, one for a color monitor, and one for a black and white monitor. Make sure that the version of **PCTE-Browser** in the *directory-containing-PCTE-Browser* is the one that is appropriate for the type of monitor being used.

3.1.3 Starting the X Window System

The PBT can be executed on any workstation running X11, release 3 or 4. This can be the Sun-supplied implementation of X, OpenWindows, or the X distribution from MIT.

It is beyond the scope of this user manual to describe how to install X, or how to start the X server on a workstation. Consult with your local system administrator.

There are a number of "standard" window managers that one may use in an X environment. The PCTE browser has been used successfully under the following such window managers: TWM (the official MIT-distributed window manager) and MWM (Motif Window Manager).

3.1.4 Starting the PCTE Server

Consult the Emeraude V12 System Administration guide for information on how to start the PCTE server.

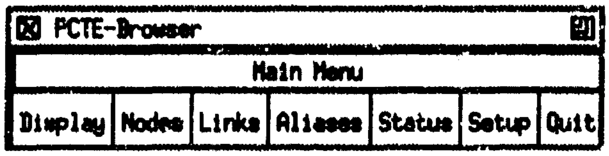


Figure 3: Main Menu Window

3.1.5 Logging into PCTE

The "standard" method for logging into PCTE is sufficient for using the PBT. Consult the Emeraude V12 System Administration guide for details.

3.1.6 Executing the PCTE Browser Tool

Once the browser has been installed in a PCTE object base, the X server, X window manager and PCTE server are executing, and the **XAPPLRESDIR** environment variable has been set, the PBT may be invoked.

3.2 BROWSER LOOK AND FEEL

The PBT application is comprised of a number of windows that are created as needed. These windows include:

- The *Main Browser* window, illustrated in Figure 3. This is the window that is opened up when the browser is first started. There is exactly one such window that remains in existence for the duration of a browser session, i.e., that exists until the browser is terminated.
- *Overall View* windows, each containing a network of PCTE nodes from the object base, as illustrated in Figure 4. This network consists of the tree of nodes formed by *Composition* links emanating from a user-specified starting point within the object base (up to a user-controllable maximum tree depth), together with, at the user's discretion, some or all of the non-composition links emanating from the nodes in this tree. The PBT can create any number of Overall View windows at the user's request. These views will remain in existence until the browser session is terminated or until the user explicitly deletes them.
- *Local View* windows, each consisting of the set of nodes that are within a user-specified distance from a user-specified starting node, together with all of the links amongst this set of nodes, as illustrated in Figure 5. The PBT can create any number of Local View windows at the user's request. As with Overall View windows, Local Views will remain in existence until the browser session is terminated or until the user explicitly deletes them.

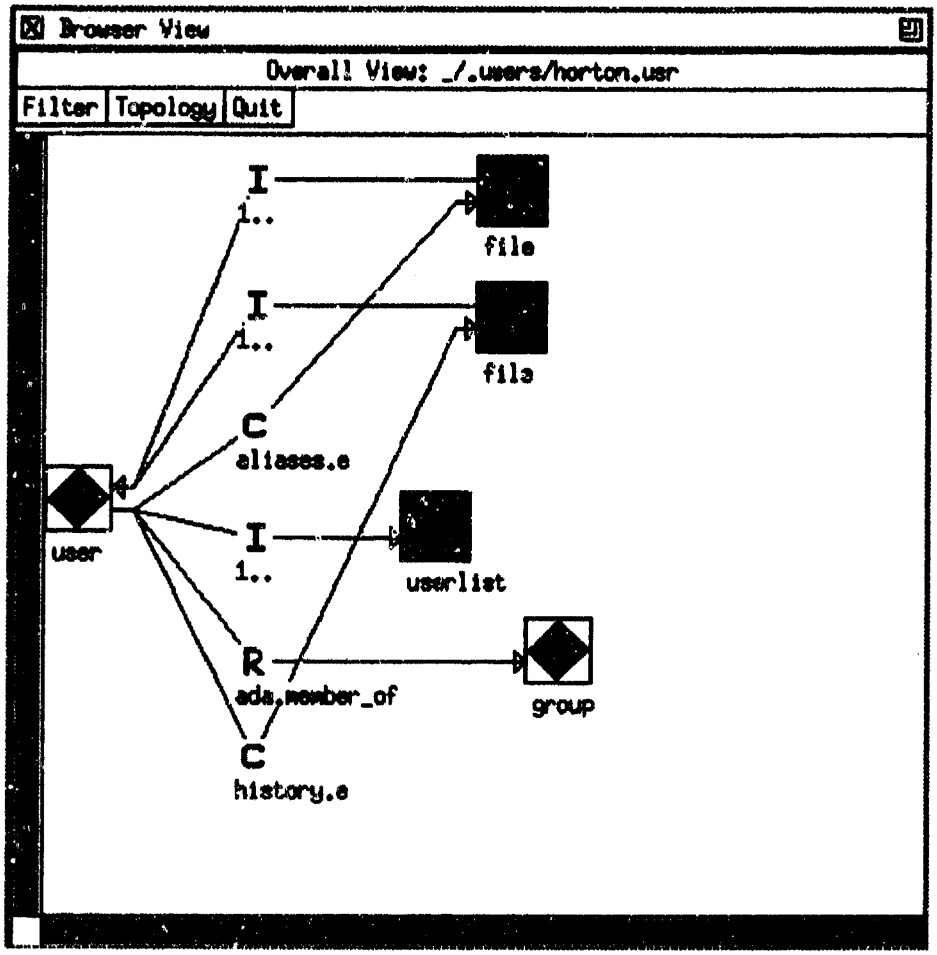


Figure 4: Overall View Window Example

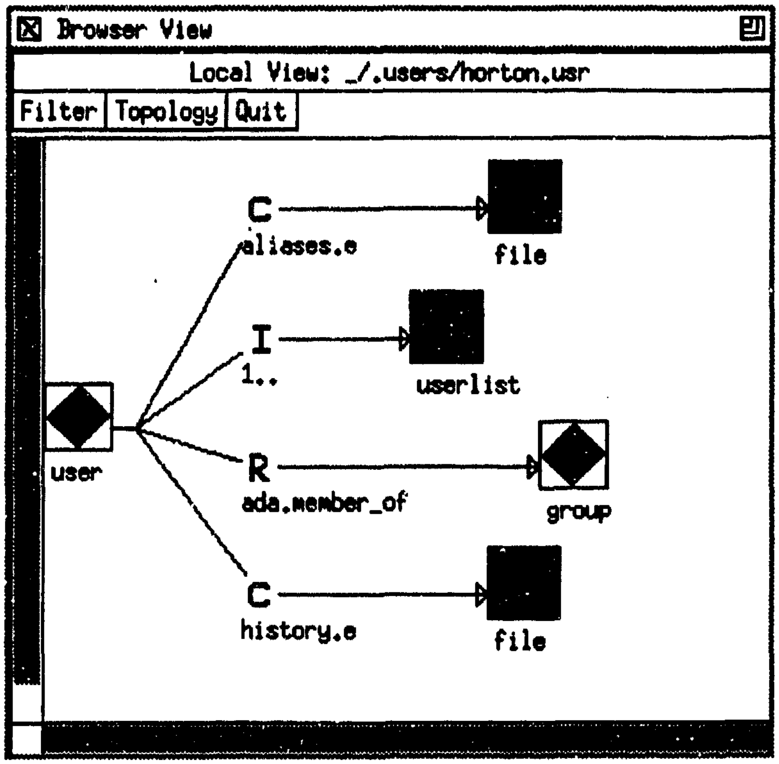


Figure 5: Local View Window Example

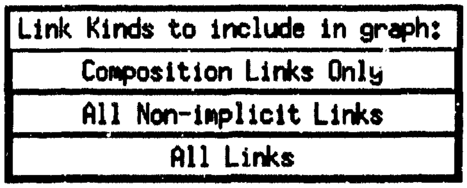
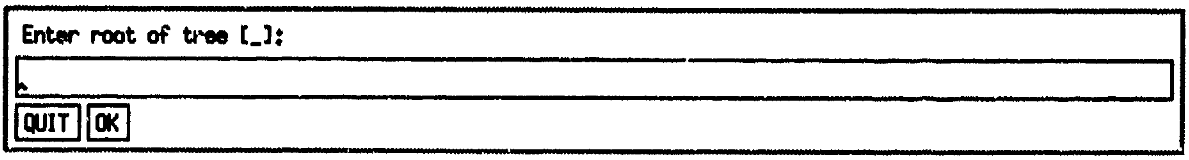


Figure 6: Menu Example

- *Menu* windows, such as the one illustrated in Figure 6, which pop up as required. For example, the menu shown in Figure 6 popped up in response to the clicking of the *Links* button in the Main Browser window. These windows close down as soon as the user selects one of the menu choices—or cancels the menu by clicking in the menu's title bar.

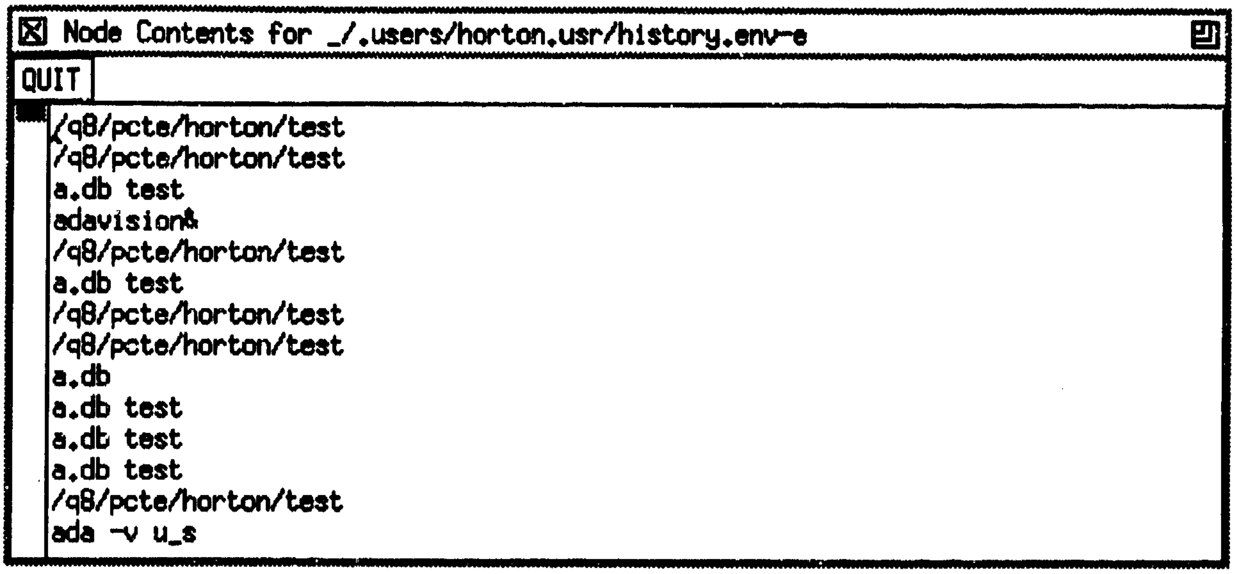
- *Dialog Boxes*, as illustrated in Figure 7. These boxes pop up when input is needed from the user, and are generally triggered by clicking on a button or by selecting a menu item. For example, the Dialog Box illustrated in Figure 7 pops up in response to clicking the `Display` button in the Main Browser window. The information needed from the user in this case is the name of the node to be used as the starting point of the tree to display in a Overall View window.

- Scrollable *Text* windows, such as the one illustrated in Figure 8. These windows are used to display PCTE File contents, the set of attributes for particular nodes or link,



```
Enter root of tree [_]:
```

Figure 7: Dialog Bcx Example



Node Contents for _/.users/horton.usr/history.env-e

```
/q8/pcte/horton/test
/q8/pcte/horton/test
a.db test
adavision&
/q8/pcte/horton/test
a.db test
/q8/pcte/horton/test
/q8/pcte/horton/test
a.db
a.db test
a.db test
a.db test
/q8/pcte/horton/test
ada -v u_s
```

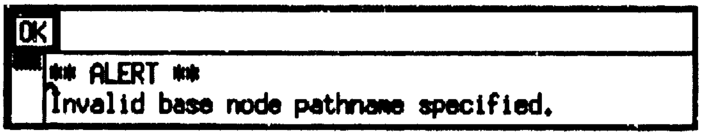
Figure 8: Scrollable Text Window Example

and for other situations requiring the display of a possibly substantial amount of text.

- *Alert Boxes*, which pop up when unexpected conditions are detected. The Alert Box illustrated in Figure 9 pops up when a pathname specified as the starting point of a network to be displayed does not exist.

3.2.1 Initiating Button Actions

As has been illustrated above, a number of PBT windows have buttons in them that can be used to initiate browser actions. Buttons are rectangular regions of the window, often



```
** ALERT **
Invalid base node pathname specified.
```

Figure 9: Alert Box Example

containing text. For example, the browser's Main Menu window shown in Figure 3 has seven buttons in it:

- Display
- Nodes
- Links
- Aliases
- Status
- Setup
- Quit

The action associated with any specific button is triggered by:

- moving the mouse cursor so that it is within the desired button, which has the effect of highlighting the button (e.g., by displaying the button's border in a thicker line or reverse videoing the button region), and then
- clicking the left mouse button once, i.e., depressing and then releasing the left mouse button while keeping the mouse cursor within the button.

If for any reason one moves the mouse cursor within a button and does not want to trigger the associated action, the user can simply move the cursor outside of the button. This can even be done if the user has depressed the left mouse button within the button, just as long as the mouse button is not released while the cursor is still within the button.

3.2.2 Selecting Menu Options

All menus created by the PBT look similar to the menu shown in Figure 6 above. There is a descriptive label at the top of the menu window, indicating the type of menu that was selected, followed by a vertical list of menu items. Such menus remain popped up until either one of the menu items is selected or the menu is "cancelled." Unlike pop up menus in some X applications, it is not necessary for the user to hold down any mouse button in order for the menu to remain open.

The menu items act as if they are buttons, meaning that the method for initiating button actions as described above applies to menu items as well. Simply click with the left mouse button while within one of the menu items.

To cancel a menu, simply move the mouse over the menu's label area and click the left mouse button. This effectively cancels any series of browser selections that led to the display of this menu.

3.2.3 Entering Text in Dialog Boxes

Figure 7 shows an example of a Dialog Box, in this case one expecting the user to type in the pathname of a PCTE object. To enter such a name, the user must first move the mouse cursor until it is inside the window's text region—the rectangular region in the center of the window—as indicated by a change of the mouse pointer. The user can then enter the text using the keyboard. Note that this text region includes its own cursor, a caret, which indicates the insertion point for new text.

3.2.3.1 Ending a Dialog

To indicate that the entered text is complete and correct, the user either hits the *Return* key or clicks the Dialog window's `OK` button (with the left mouse button).

To cancel the operation that caused the Dialog window to pop up, the user must click the Dialog window's `QUIT` button.

3.2.3.2 Making Changes To Entered Text

The following options exist for the user if corrections or changes need to be made to the text being typed into the Dialog window:

The `Delete` key deletes the character to the left of the text cursor, as will the `Back Space` key (or `ctrl-H`).

`Ctrl-B` and `ctrl-F` move the text cursor one character to the left or right respectively. This allows the user to insert or delete text in the middle of existing input.

`Ctrl-A` moves the text region's cursor to the beginning of the line of text, i.e., to the extreme left end of the text.

`Ctrl-E` moves the text region's cursor to the end of the line of text, i.e., to the extreme right end of the text.

`Ctrl-K` deletes all the text from the cursor position to the end of the line.

3.2.4 Scrolling Within View Windows

The Overall and Local View windows are used to display networks of PCTE nodes—many of which may be too large to fit entirely within the View window. What is displayed, therefore, may be just one section of the complete network.

3.2.4.1 Use of View Scroll Bars

The primary method of moving around the network is via the scroll bars that appear at the left and bottom of each View, using the "standard" X conventions for scrolling. That is:

Clicking the left mouse button while the mouse pointer is within the left scroll bar moves the section of the network being displayed "forward." i.e., further down in the network.

Clicking the right mouse button while the mouse pointer is within the left scroll bar moves the section of the network being displayed "backward."

Clicking the middle mouse button while the mouse pointer is within the left scroll bar "jumps" to a different section of the network, dependent upon where (proportionally) in the scroll bar the middle button was clicked.

Similarly, the section being displayed can be scrolled (or "jumped") to the right or left by clicking in the scroll bar at the bottom of the window with the left, right or middle mouse button.

3.2.4.2 Use of View's Topology Region

A secondary method for moving the section of the network being displayed involves the use of the Topology region of the View, as illustrated above in Figure 2. This region is not part of the View window when first created, but can be added to the View at any time (as described in section 4.3.3 below).

The darkened area within the Topology region identifies the portion of the whole network that is being displayed in the left ("Main") region of the View window. If the left mouse button is clicked anywhere in the Topology region the section of the network shown in the Main region will be moved such that the location in the network at the place clicked will be displayed—centered in the Main region if possible. As a side-effect, the darkened area within the Topology region will move as well to reflect what is displayed in the Main region.

Note that the Topology region may not be big enough to display the entire network, even in miniaturized form, as is shown in Figure 2. In such cases, the Topology region's own scroll bar(s) may be used to scroll around the Topology region. This scrolling is done identically to scrolling the main view as described in section 3.2.4.1.

3.2.5 Scrolling Within Text Windows

Text windows, such as the one shown in Figure 8, are used to display the contents of File nodes. (See section 4.7.3 below for a description of how to create them.) A Text window is created with a scroll bar at the left. This allows the user to click with the left or right button in the scroll bar to move up or down, respectively, or the click with the middle button to jump around in the text.

3.2.6 Miscellaneous Window Manager Operations

A number of functions related to the management of browser windows are controlled by the X window manager in use at the time the browser is invoked (e.g., MWM or TWM). These functions include:

- Resizing windows
- Iconifying windows
- Placing windows on the screen and moving them around the screen
- Exposing browser windows covered by other windows
- Hiding browser windows behind other windows

The means by which these operations are accomplished depends upon which window manager is in use. These window managers decorate most browser windows with title bars with "buttons" (mouse-sensitive regions) that can be used to initiate some or all of the above functions. All of the examples illustrated in this document were created using TWM, the MIT-distributed window manager. Consult the documentation for the specific window manager in use for explanations as to how the title bar buttons are used, and for alternate methods for initiating window manager functions.

3.2.7 Eliminating Browser Windows

Some windows created by the browser are, by design, of short duration, namely pop up Menus and Dialog Boxes. These windows require "immediate" user attention, i.e., until the user completes the action associated with these windows—selects the menu item or enters the text into the dialog box—no additional PBT functions can be invoked. Once these short-duration windows have served their purposes, they automatically disappear.

All other PBT windows—the Main Menu, Views, Text and Alert Boxes—remain in existence for as long as the user wishes.

Each of the "longer-duration" windows can be iconified by the X Window Manager as discussed in section 3.2.6 above. However, the PBT itself is responsible for removing (destroying) these windows, providing a `Quit` button in each such window to do so.

Clicking the `Quit` button in any View, Text or Alert Window will destroy that window, but will leave all other browser windows intact.

3.2.8 Terminating the Browser Session

Clicking the `Quit` button in the Main Menu window destroys all browser windows created in that PBT session, and terminates the browser.

4 PCTE Browser Capabilities

The subsections below describe the following PBT capabilities:

- PCTE object identification
- name pattern conventions
- creation/maintenance of View windows
- filtering of nodes within View windows
- filtering of links within View windows
- operations on nodes
- operations on links
- display of error status information
- exiting the browser

4.1 Object Identification

Several PBT functions prompt the user to identify the object in the PCTE object base for which the function is to be applied.

The PBT recognizes standard PCTE pathnames starting either from the root ("_") or the user's home ("~"). Examples of such pathnames include:

```
-
~/history.e
_/.users/guest.usr
```

Relative pathnames are those beginning with a link_name, such as: **`history.e`**. Such pathnames are interpreted within the PBT as relative to the PBT session's *Current Object*. For example, if **`_/.projects`** was the set to be the Current Object earlier in the PBT session, then the previous relative pathname would be interpreted as **`_/.projects/history.e`**. (See section 4.6.1 for information on how to set the Current Object.)

4.1.1 Aliases in Object Identification

As a final shorthand method for object identification, the PBT supports the creation and maintenance of a set of aliases whose values are the object pathnames, and allows objects to be identified via the use of these aliases.

Assume for the moment that one such alias already exists, **path_ref**, and assume that the values of this alias is "**_/.users/guest.usr**". Then the following are valid examples of object identifications:

```
$path_ref
$path_ref/history.e
```

References to the values of object aliases always begin with a dollar sign ("$") followed immediately by the name of the object alias. The PBT interprets such references by substituting the *$alias_name* by the current value of the alias. This means that these references are equivalent to:

```
_/.users/guest.usr
_/.users/guest.usr/history.e
```

The PBT is case-sensitive; i.e., the case used when specifying the name of the alias is significant.

The PBT maintains the set of aliases via the menu options associated with the **Aliases** button on the Main Menu (see Figure 3). Clicking this button causes the **Aliases** menu to pop up, as shown in Figure 10. These options allow one to:

- display the current set of aliases
- create new aliases or modify existing ones
- delete existing aliases
- load alias names and values from existing PCTE Files
- save the existing set of aliases in PCTE Files

Clicking on the **Display Aliases** option pops up a Text window that contains the current set of object aliases, as illustrated in Figure 11. The aliases are displayed one per line, as in:

Alias Name: *Alias Value*

Note that **Current_Object** is one of the aliases displayed in Figure 11. This is a consequence of the fact that the PBT treats the Current Object internally as an object alias, rather than relying upon the underlying PCTE implementation to manage it.

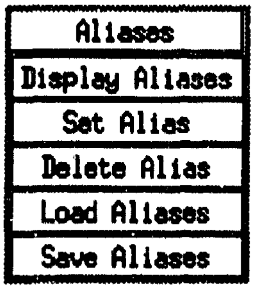
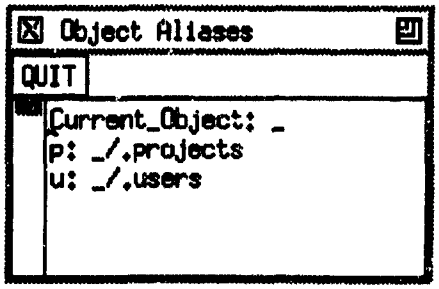


Figure 10: Aliases Menu



```
Current_Object: _
p: _/.projects
u: _/.users
```

Figure 11: Display Aliases Text Window

Clicking on the Set Alias option causes a Dialog Box to pop up, as shown in Figure 12. The user can then enter the name of an alias. When the user clicks in this Dialog Box's Ok button, the Dialog Box shown in Figure 13 pops up. The user can then enter the value, i.e., the PCTE pathname, associated with that alias. If the previously specified alias name doesn't yet exist, it will be added to the set of aliases; if it already exists, then its value will be modified.

Clicking on the Delete Alias option causes a Dialog Box to pop up, as shown in Figure 14. By default, all existing aliases (names and values) will be deleted. If the user enters a specific alias name, then only that alias will be deleted.

Clicking on the Save Aliases option causes a Dialog Box to pop up, as shown in Figure 15. This allows the user to identify the PCTE File in which to save the current set of node aliases. If this File already exists, its contents will be overwritten by this list of aliases. Otherwise, the PBT will create this File. (NOTE: The one alias that is not saved in this operation is the current value of Current_Object.)

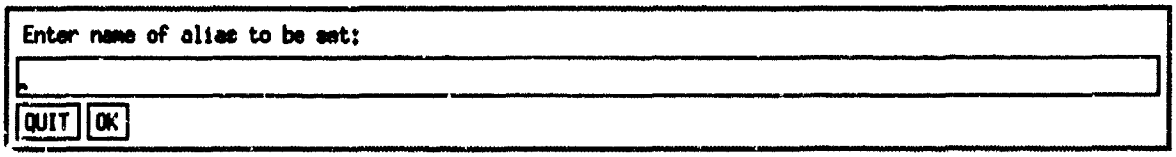

Enter name of alias to be set:

Figure 12: Set Alias Name Dialog Box

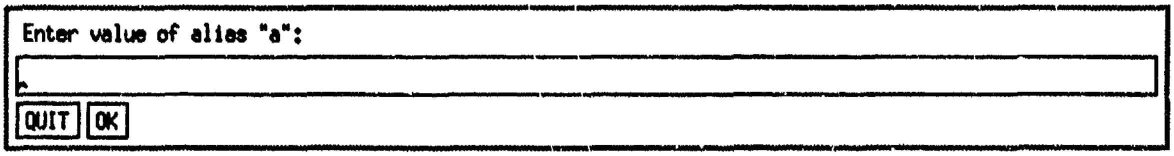

Enter value of alias "a":

Figure 13: Set Alias Value Dialog Box

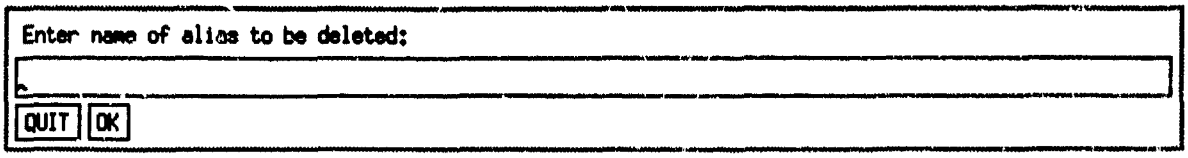

Enter name of alias to be deleted:

Figure 14: Delete Alias Dialog Box

The format used for saving the node alias names and values in the File node is exactly the same as that used for displaying them, as shown in Figure 11. It is also the same format expected by the `Load Aliases` option discussed below.

Clicking on the `Load Aliases` option causes a Dialog Box to pop up, as shown in Figure 16. This allows the user to identify a PCTE File node containing a set of node aliases to be used for filtering purposes. If any aliases already exist at the time a new set is loaded, the new set is merged with the existing set.

4.2 Name Pattern Syntax

Several PBT functions prompt the user to enter the name of a link or object type name, or a name *pattern* that will match one or more such names. The following rules apply whenever such names or patterns are required:

- These names are assumed to be made up of letters, digits, underscores and/or hyphens. All such characters in name patterns match exactly the same character; that is, these names are case-sensitive.
- In addition to the alphanumeric, hyphen and underline characters that can actually comprise these names, two special characters can also be used:
  - "*" matches *any* string of zero or more characters.

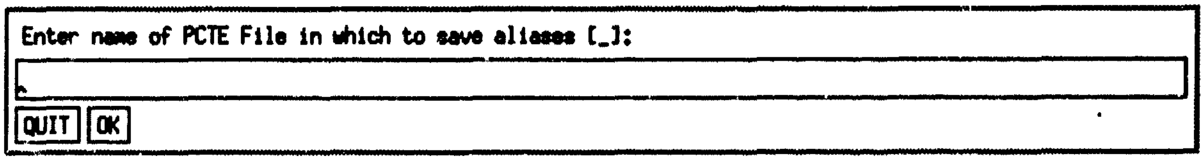
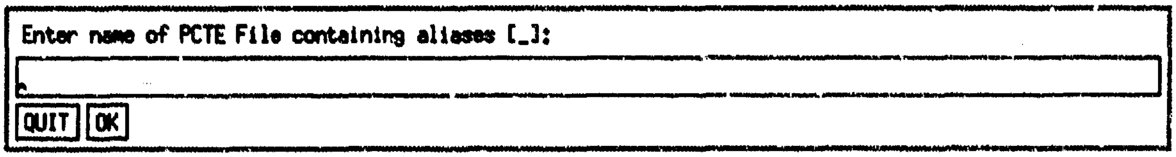

Enter name of PCTE File in which to save aliases [_]:

Figure 15: Save Aliases Dialog Box



Enter name of PCTE File containing aliases [_]:

Figure 16: Load Aliases Dialog Box

- "?" matches any single character.

4.3 Creation/Maintenance of View Windows

There are two types of View windows that one can create using the PBT:

- Overall Views contain a tree of nodes formed by the *Composition* links from a given starting object, optionally including the other types of links emanating from the nodes of this tree.
- Local Views contain the set of objects that are within a user-specified distance of a starting object, together with all types of links amongst these objects.

The subsections below describe how these two types of Views are created. How the Topology region—"the view from one thousand feet"—can be added to or removed from such Views is also described.

4.3.1 Creation of Overall View Windows

In order to create an Overall View window, one must first identify the PCTE object that is to be the root of the Composition Tree. This can be done in one of two ways.

The first way of identifying the root object is by entering its pathname:

1. Click the `Display` button in the Main Menu window (with the left mouse button). This will pop up a Dialog window as shown in Figure 17.
2. Enter the pathname of the object to be used as the root of the tree. See section 4.1 for a discussion of allowable pathnames. (The Current Object value is the default for this operation.)
3. When finished typing the pathname, either hit the *return* key or click the `OK` button to continue, or the `QUIT` button to cancel the operation.

The second method is via a pure point-and-click operation:

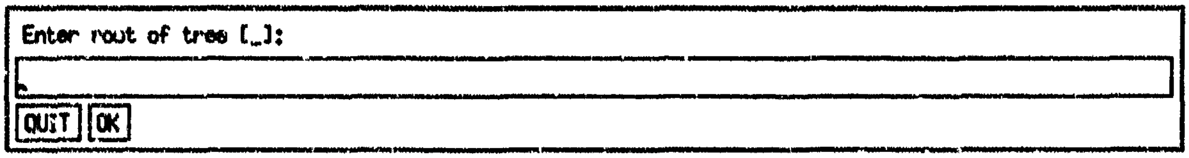


Figure 17: Dialog Box for Overall View Starting Point Pathname

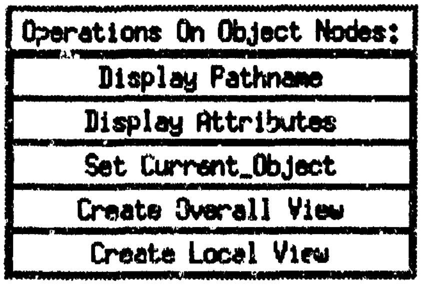


Figure 18: Node Operations Menu

1. Click on any node icon in any existing Overall or Local View window. This will pop up a menu such as that shown in Figure 18.
2. Click in the `Create Overall View` menu item.

Regardless of how the root object was selected, the following steps must be taken to complete the creation of the Overall View:

1. The `Link Kinds to include in graph` Menu window, shown in Figure 19, will pop up. Click the appropriate menu item to either select Composition links only, all but Implicit links, or all types of links. To cancel the operation at this point, click in the Menu window's title bar.
2. If the operation has not been cancelled, the PBT will complete the operation (which can take a while, depending upon how complex the network is). When the View is ready for display, the user chooses where to place the window by moving the mouse and clicking the left mouse button. The PBT will then be ready to perform another user-selected operation.

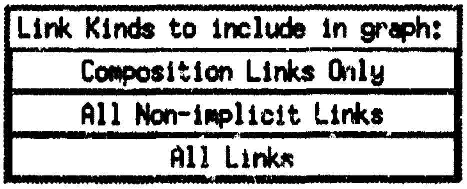
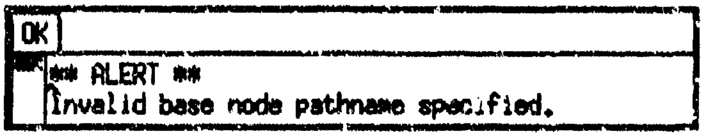


Figure 19: Link Kind Menu



** ALERT **
Invalid base node pathname specified.

Figure 20: Alert Box for Pathname Problems

The maximum depth of the Composition Tree within an Overall View is under user control. In the current PBT release, this maximum depth is currently considered to be global to the PBT session, rather than being asked for in each Overall View creation operation. The default maximum depth is 2. See section 4.9 for instructions on how this value can be modified during a PBT session.

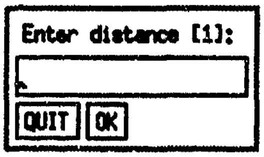
Should the browser detect any problem with the network, most likely caused by the user's specification of a syntactically illegal pathname, or of a non-existent node, then an Alert Box similar to that shown in Figure 20 will be displayed instead of the View window itself. Click the OK button in the Alert Box when finished with the message.

NOTE: As is true for all PCTE tools, the PBT is constrained by all the rules governing PCTE access control. This means that objects which the PBT user is not entitled to access will not be included in browser Views.

4.3.2 Creation of Local View Windows

In order to create a Local View window, one must do the following:

1. Click the left mouse button while pointing at any node icon in an Overall or Local View window. This will bring up a menu similar to Operations on Nodes Menu window shown in Figure 18.
2. Click on the Create Local View option. This will pop up a Dialog window as shown in Figure 21. To cancel this operation, click on the Menu window's title bar instead.
3. Enter the number to use for the maximum distance from the starting node in the Dialog window. If not specified, the default is 1.
4. When finished entering the distance, click the OK button to continue, or the QUIT button to cancel the operation.
5. If the operation has not been cancelled, the PBT will complete the operation (which can take a while, especially if the distance is greater than 1). When the View is ready for display, the user chooses where to place the window by moving the mouse and clicking the left mouse button. The PBT will then be ready to perform another user-selected operation.



Enter distance [1]:

Figure 21: Local View Distance Limit Dialog Box

4.3.3 Creation of Topology Regions in View Windows

Section 3.2.4.2 described how the Topology region of a View window, such as the one displayed at the right side of Figure 2, can be used to move around a large network. Such a region can be added to or removed from any Overall or Local View window simply by clicking in the `Topology` button in any View window. If the Topology region had not been part of the View window prior to clicking this button, then the region will be added to the window; if this region already was being displayed, then clicking in the `Topology` button causes it to be removed.

4.4 Hiding/Revealing Nodes and Links Within View Windows

Once a set of objects has been displayed in a Local or Overall View window, the PBT allows the user to restrict this set to a specific subset of basic object types. That is, the user is able to direct the PBT to redraw the network of objects, filtering out selected objects, specifying them either by basic kind (e.g., File or non-file Object), or by the definition names associated with the objects—the names used to label the objects within the Views; such filtering of objects also hides those links originating and/or terminating at the filtered out objects. In addition, the user can specify a set of links, either by kind (e.g., *composition* or *reference*) or by link name, and the PBT will redraw the network without including such links.

This filtering of the objects and/or links in an existing View is performed by first clicking the `Filter` button in any View window. This will pop up the `Filter` Menu as shown in Figure 22. After this, the type of hiding/revealing that takes place depends upon the menu option selected.

4.4.1 Hiding/Revealing Nodes Based Upon Object Kinds

If the `Suppress Modes By Kind` Menu option from the Filter menu (shown in Figure 22) is selected, the PBT will pop up a second menu shown in Figure 23. The user can then choose to hide any single kind of object, or to hide *all* object kinds. (The reader may note that in anticipation of a conforming ECMA PCTE environment, PROCESS is one kind of object that is referred to, even though, in the PBT implementation on top of Emeraude V12, such objects do not yet exist.)

Similarly, if the `Unsuppress Nodes By Kind` menu option from the Filter menu is selected,

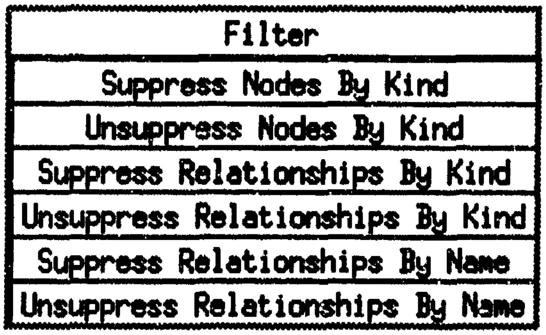

Filter
Suppress Nodes By Kind
Unsuppress Nodes By Kind
Suppress Relationships By Kind
Unsuppress Relationships By Kind
Suppress Relationships By Name
Unsuppress Relationships By Name

Figure 22: Filter Menu

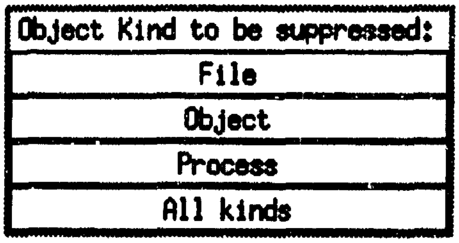


Figure 23: Suppress by Object Kind Menu

the PBT will pop up the menu shown in Figure 24. The user can then choose to reveal any single type of object, or to reveal *all* object kinds.

4.4.2 Hiding/Revealing Links Based Upon Link Type

If the **Suppress Links By Kind** menu option is selected, the PBT will pop up the menu shown in Figure 25. The user can then choose to hide selected kinds of links. (The reader may note that in anticipation of a conforming ECMA PCTE environment, EXISTENCE and DESIGNATION link kinds are referred to, even though, in the PBT implementation on top of Emeraude V12, such link kinds do not yet exist.)

Similarly, if the **Unsuppress Links By Kind** menu option is selected, the PBT will pop up the menu shown in Figure 26. The user can then choose to reveal elected kinds of links.

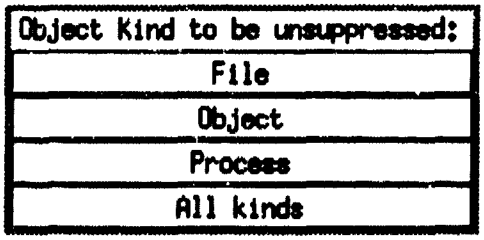


Figure 24: Unsuppress by Node Kind Menu

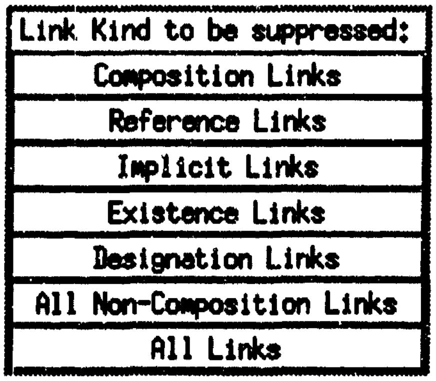


Figure 25: Suppress by Link Kind Menu

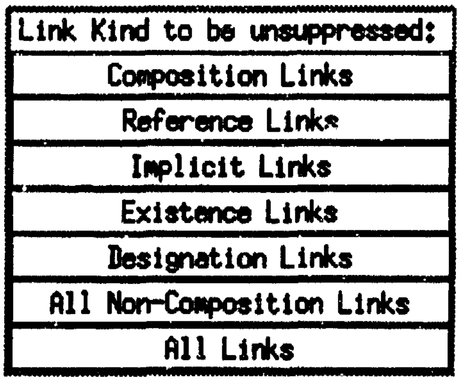


Figure 26: Unsuppress by Link Kind Menu

Note that the removal of links from the graph does not affect the set of nodes being displayed; i.e., nodes will be displayed even if they are totally disconnected from the rest of the nodes in the graph.

4.4.3 Hiding/Revealing Nodes Based Upon Object Type Name

If the **Suppress Nodes By Name** menu option is selected, the PBT will pop up a Dialog Box shown in Figure 27. The user can then type the name of node to be hidden, or a name pattern describing a set of such names (as defined in section 4.2 above) to be hidden as a group. These names are the ones used to label the various nodes within View windows, e.g., common_root, toolset, group, user, etc. As with the filtering of nodes by more general node types, discussed in section 4.4.1 above, links emanating or terminating at hidden nodes will also be hidden.

Similarly, if the **Unsuppress Nodes By Name** menu option is selected, the PBT will pop up a Dialog Box shown in Figure 28. The user can then type the name of object type to be revealed, or a name pattern describing a set of such names.

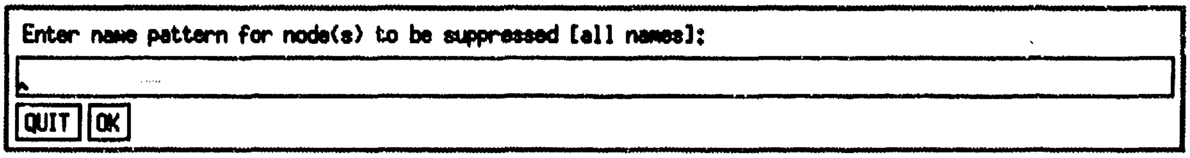

Enter name pattern for node(s) to be suppressed [all names]:

Figure 27: Suppress Node Name Dialog Box

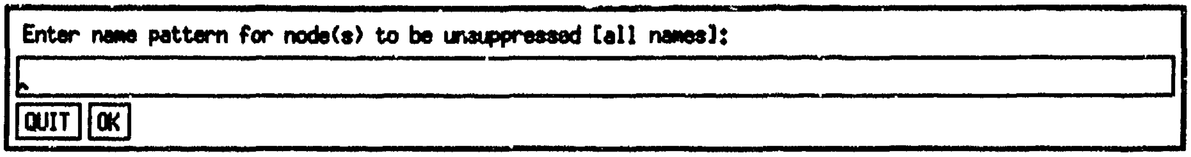

Enter name pattern for node(s) to be unsuppressed [all names]:

Figure 28: Unsuppress Node Name Dialog Box

4.4.4 Hiding/Revealing Links Based Upon Link Name

If the `Suppress Links By Name` menu option is selected, the PBT will pop up a Dialog Box shown in Figure 29. The user can then type the name of relation to be hidden, or a name pattern describing a set of such names (as defined in section 4.2 above) to be hidden as a group.

Similarly, if the `Unsuppress Links By Name` menu option is selected, the PBT will pop up a Dialog Box shown in Figure 30. The user can then type the name of relation to be revealed, or a name pattern describing a set of such names.

4.5 A Priori Filtering of Links

The previous section described how one can remove nodes and/or links from an existing graph in a View in order to focus on what may be most important to a user at any given moment. This section discusses a PBT capability that allows the user to prune nodes and links from a graph before the network is ever displayed.

The user can optionally create and maintain a list of link name patterns (e.g., `*list`). If such a list of patterns has been specified at the time that a Local or Overall View is constructed, the PBT will ignore any link that does not match at least one of the specified patterns when it creates the View.

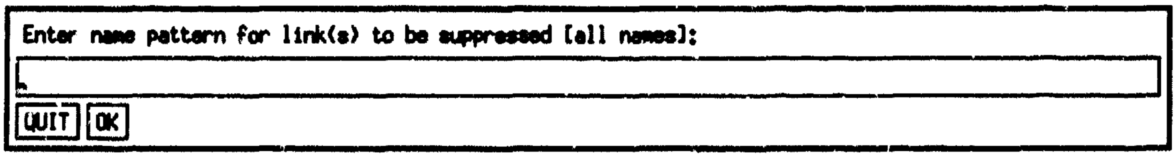
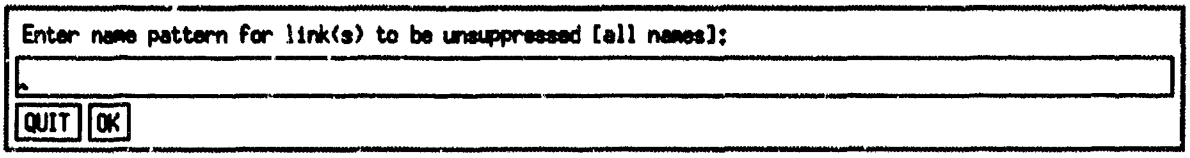

Enter name pattern for link(s) to be suppressed [all names]:

Figure 29: Suppress Relation Name Dialog Box



Enter name pattern for link(s) to be unsuppressed [all names]:

Figure 30: Unsuppress Relation Name Dialog Box

Any existing, non-empty list of link name patterns affects the construction of Overall Views in the following manner: If any of the primary relations that make up the tree of nodes rooted at the specified starting node do not match any of the listed patterns, then the entire subtree of nodes "below" that unmatched primary link will be omitted from the constructed graph. In addition, any secondary links emanating from any nodes in the "reachable" portion of the primary tree, will, likewise be excluded from the constructed graph.

Any non-empty list of link name patterns affects the construction of Local Views in the following manner: Nodes will be included in the constructed graph only if they are within the specified distance from the starting node following only those links whose names match at least one of the specified patterns. Also, the only links included in the constructed graph will be those matching one of the specified patterns.

It is important to note that the filtering/unfiltering of nodes and links discussed in the previous section apply only to those nodes and links that are included in the initially constructed View. That is, it is not possible to add nodes and/or links that were excluded from the graph initially without modifying (or eliminating) the list of link name patterns and "recomputing" the network.

The user is given the ability to manage this list of name patterns via the menu options associated with the `Links` button on the Main Menu (see Figure 3). Clicking this button pops upthe `Links` menu, as shown in Figure 31. These options allow one to:

- display the current set of link name patterns
- specify new patterns
- delete existing patterns
- load patterns from existing PCTE File nodes
- save the existing set of link name patterns in PCTE File nodes

Clicking on the `Display Link Name Patterns` option pops up a Text window that contains the current list of name patterns, as illustrated in Figure 32.

Clicking on the `Add Link Name Pattern` option causes a Dialog Box to pop up, as shown in Figure 33. The user can then enter a new Link Pattern to be added to the current set of such patterns.

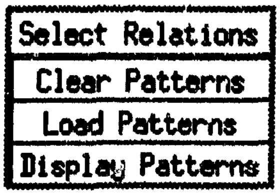
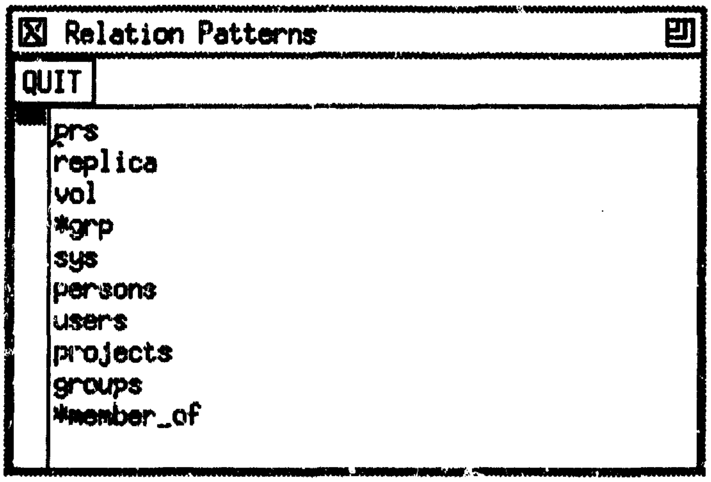


Figure 31: Links Menu



```
prs
replica
vol
*grp
sys
persons
users
projects
groups
*member_of
```

Figure 32: Display Patterns Text Window

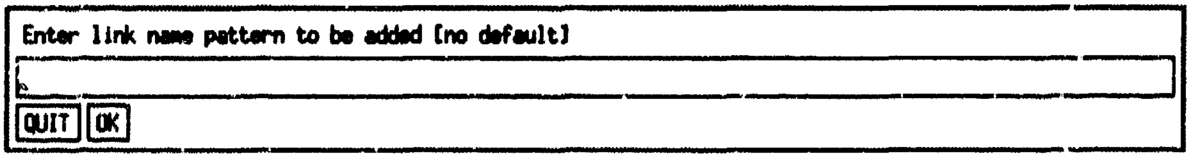

Enter link name pattern to be added [no default]

Figure 33: Add Pattern Dialog Box